EE599 Project-Landscape image inpainting based on GAN

Zhenye Jiang, Yanbang Kan, Maria Mangassarian

MAY 7, 2019
USC

Report Outline

1. Project objectives

The objective of our project is to build a GAN which can inpaint landscape images with a missing area. The landscape picture will be covered by a square blank mask and our model will recover the original picture by making the missing area as real and natural as possible, as well as contextually consistent with the rest of the picture. We were able to accomplish this by taking as reference an existing model that does this task in [1] and by proposing and evaluating modifications to that model in order to get the results we desire.

In addition, through our research on the existing GAN models and our work on building a GAN, our objective is to gain more understanding as well as experience on GANs. GAN being a relatively new and exciting topic in deep learning, we were able to further build on the knowledge on GANs presented in class and learn first-hand all the challenges that come with training a GAN. Furthermore, we got familiar with cutting edge technologies and applications regarding deep learning while in the process of choosing our project topic.

2. Prior work and model architecture

In paper [1], it mentioned a model that worked well in inpainting. The author involved training two networks which are completion network (generator) and discriminator. As it is shown below, the model for the paper is the one that we are referring.

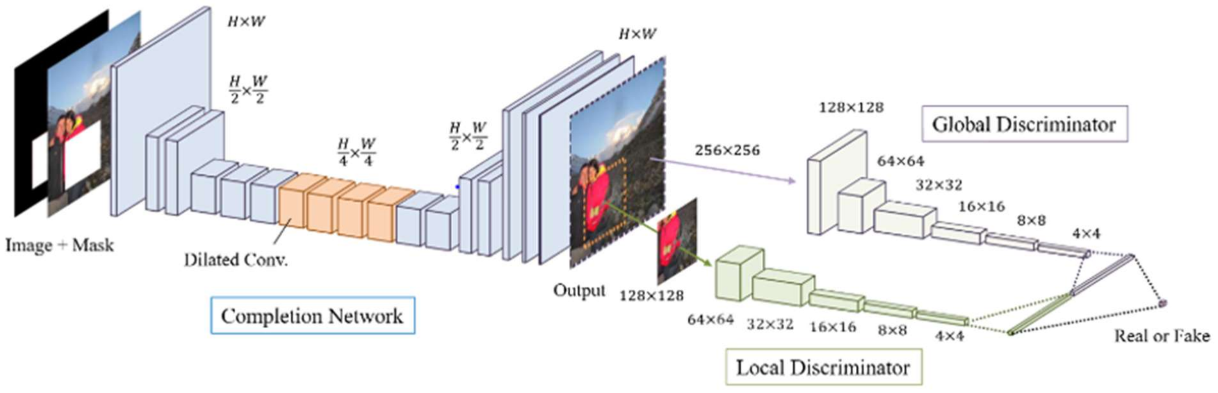


Figure 1[1]

As the figure shows, the original image will be applied with the mask and later the 3 channels RGB image with missing part will be taken along with the mask as the whole input to the generator network. Briefly, the input image data will be processed by convolutional layers in CNN to be resized to quarter of the original size. Specifically, the data will go through 4 dilated convolutional layers which help the network gain a better receptive field of the picture as a whole. In other words, the kernel is applying on more pixels far away from center to enable the network to see more. Lastly the completion network will restore the size of the processed data.

For more about the dilated layers: rather than applying normal convolutional layers, the dilated layer allows increasing the number of learnable weights by spreading the kernel across the input map. To be specific, if one 2D layer is a C-channel h × w map and the next layer is a C′-channel h′ ×w′ map, the dilated convolution operator can be written for each pixel as:

$$\mathbf{y}_{u,v} = \sigma\left(\mathbf{b} + \sum_{i=-k'_h}^{k'_h} \sum_{j=-k'_w}^{k'_w} W_{k'_h+i,k'_w+j}\ \mathbf{x}_{u+\eta i, v+\eta j}\right),$$
$$k'_h = \frac{k_h - 1}{2}, \qquad k'_w = \frac{k_w - 1}{2},$$

Figure 2[1]

The kw and kh are the kernel width and height (odd numbers), η is the dilation factor, $x_{u,v} \in R^C$ and $y_{u,v} \in R^{C'}$ are the pixel component of the input and the output of the layer, $\sigma(\cdot)$ is a component-wise non-linear transfer function, $W_{s,t}$ are C′-by-C matrices of the kernel, and $b \in R^{C'}$ is the layer bias vector. Specifically, when $\eta = 1$ the equation returns to the standard convolution operation.

For discriminator part, the original model contains two discriminators which are global discriminator and local discriminator. They are considered together to improve the contextual consistency and the visual quality of the generated pictures. Also, as it is known that GAN is difficult to train, this process of taking two discriminators into consideration helps in increasing the stability of the network. The filled part generated by the completion network and the full-size generated image will be served as the input to local and global discriminator respectively. In the end for classification, the two part will be resized, flatten and then combined as the input of the fully connected network (more details later).

Comparing this architecture with 2 previous proposed method Patch-based and Context encoders, as the author concluded, the results can be found in the table below:

	Patch-based	Context encoder	Ours
Image size	**Any**	Fixed	**Any**
Local Consistency	**Yes**	No	**Yes**
Semantics	No	**Yes**	**Yes**
Novel objects	No	**Yes**	**Yes**

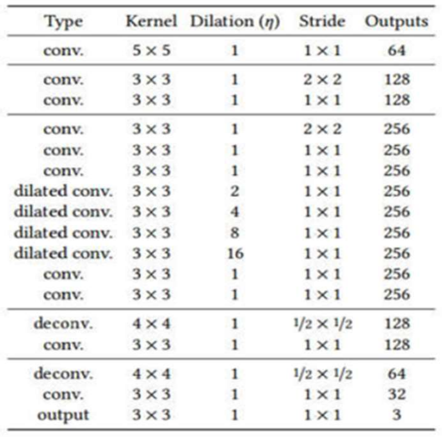
Table 1[1]

The new architecture is able to show high quality instructions for arbitrary images size and masks. Moreover, it not only shows the local consistency but also shows high semantic understanding of the image. In addition, it is able to generate novel content comparing to Patch-based network and the novel content has a higher resolution than the one generated by Context encoders.

For training process, the approach of training the network is that the tree networks: completion network, global and local discriminator networks are trained together. During the training, the completion network is trying to fool the discriminators while the two discriminators are trying to determine whether the image has been completed.

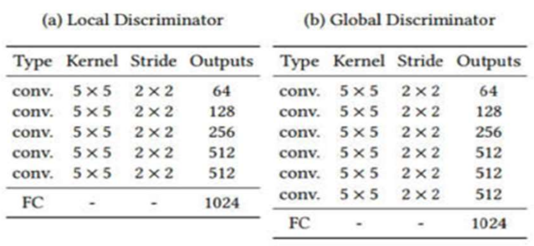
According to the description above, the table given below shows more details of settings of the original model used in paper [1].

Table 1: Architecture of completion network



Type	Kernel	Dilation (η)	Stride	Outputs
conv.	5×5	1	1×1	64
conv.	3×3	1	2×2	128
conv.	3×3	1	1×1	128
conv.	3×3	1	2×2	256
conv.	3×3	1	1×1	256
conv.	3×3	1	1×1	256
dilated conv.	3×3	2	1×1	256
dilated conv.	3×3	4	1×1	256
dilated conv.	3×3	8	1×1	256
dilated conv.	3×3	16	1×1	256
conv.	3×3	1	1×1	256
conv.	3×3	1	1×1	256
deconv.	4×4	1	$1/2 \times 1/2$	128
conv.	3×3	1	1×1	128
deconv.	4×4	1	$1/2 \times 1/2$	64
conv.	3×3	1	1×1	32
output	3×3	1	1×1	3

Table 2: Architecture of Discriminator



(a) Local Discriminator

Type	Kernel	Stride	Outputs
conv.	5×5	2×2	64
conv.	5×5	2×2	128
conv.	5×5	2×2	256
conv.	5×5	2×2	512
conv.	5×5	2×2	512
FC	-	-	1024

(b) Global Discriminator

Type	Kernel	Stride	Outputs
conv.	5×5	2×2	64
conv.	5×5	2×2	128
conv.	5×5	2×2	256
conv.	5×5	2×2	512
conv.	5×5	2×2	512
conv.	5×5	2×2	512
FC	-	-	1024

Table 2[1]

The proposed network from paper [1] is actually based on the CNN network which extract the features and convert the information to vectors. Output of the networks are combined by a concatenation layer that predict the probability of the image being real. As shown in the table above, the network consists of 6 convolutional layers and a single fully-connected layer (the output size is a 1024-dimensional vector) and etc.

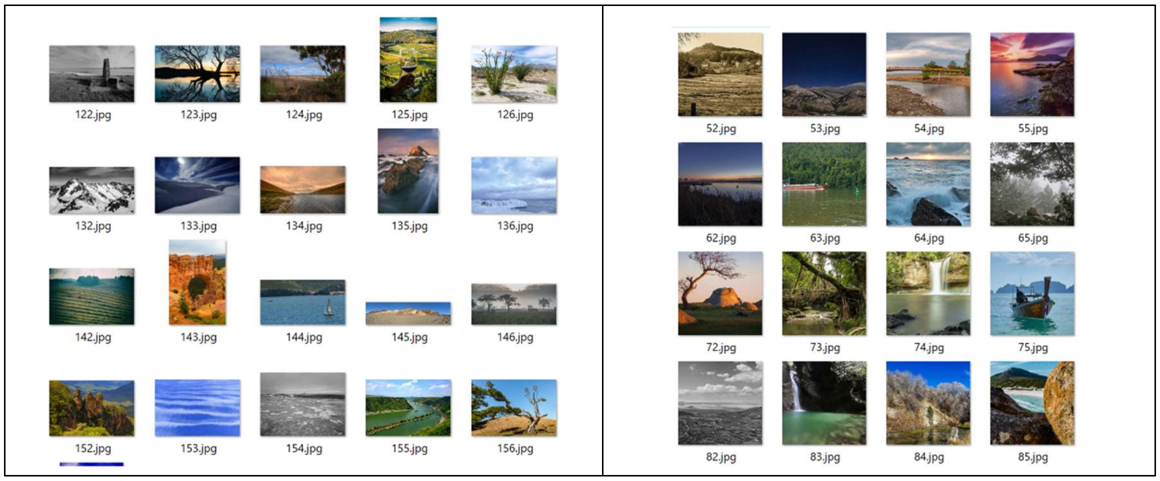
Based on the model mentioned above, we made some changes to make the new model fit to the project. Basically, the generator setting is the same according to layers design but MSE loss is instead used for generator training.

And for discriminator, rather than considering two we only keep the local discriminator. Moreover, instead of doing classification we use the real and fake partial image to calculate the Wasserstein distance[6] as the loss function. The goal of the process is to try to fit the real context of the image.

3. Dataset collection:

In this part, we downloaded 145K photos from Flickr with commercial use allowed license. The original photos size is not smaller than 350px in height or width. Our generator is a pixel2pixel architecture so that it is compatible to all resolution input images, but we crop these images into 256x256 resolution for the convenience of batch training.

Due to the size of dataset is quite large for this one-month project. In the training procedure, we only chose 96000 images as our training dataset, and all experiment processes mentioned below is on that small dataset.



4. Modifications and Problem occurred

a. Modification of loss

The original paper we referred to was using DCGAN to train their deep networks. However, we read some papers mentioned that DCGAN is hard to train. It requires many hyper-parameters adjustment during the training process and also needs the distributions of generated images and real images have common crossed areas. That means their common distributions measurement should be bigger than 0 which is quite hard to satisfied. Also, DCGAN loss cannot represent the process of training. Therefore, we turned to WGAN-GP to avoid these problems.

Different from DCGAN loss for Discriminator which is trying to separate generated images and real images, WGAN-GP loss defines a measurement named Wasserstein distance to represent the distribution distance between generated images and real images and tries to minimize it. Therefore, the discriminator loss of WGAN-GP represents the similarity of generated images and real images.

$$\mathrm{MSE}_{\mathrm{loss}} = \mathrm{E}((\mathrm{G}(\mathrm{x}) - \mathrm{x})^2)$$
$$\mathrm{D\_loss} = \mathrm{E}(\mathrm{D}\big(\mathrm{G}(\mathrm{x})\big) - \mathrm{E}\big(\mathrm{D}(\mathrm{x})\big) + \mathrm{grad}_{\mathrm{pen}} + 10 * \mathrm{MSE}_{\mathrm{loss}}$$
$$\mathrm{G\_loss} = -\mathrm{E}(\mathrm{D}\big(\mathrm{G}(\mathrm{x})\big) + 10 * \mathrm{MSE}_{\mathrm{loss}}$$

b. Modification of training process

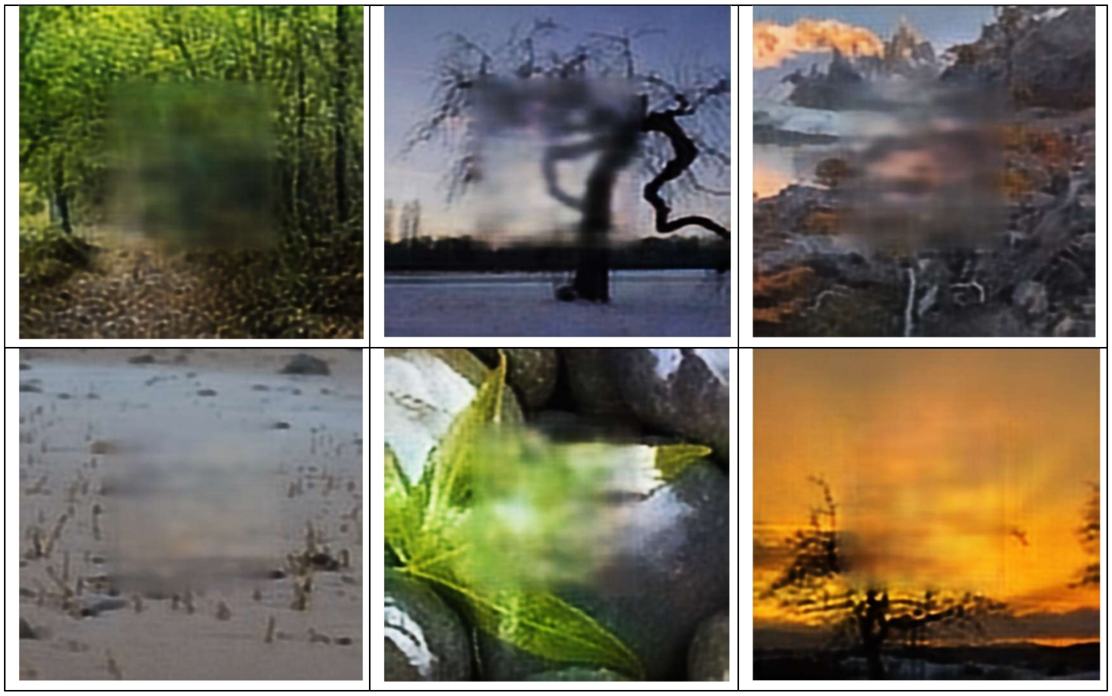
When training the network, we firstly trained discriminator 3 times and generator once, and find that the loss of discriminator diverged quickly. Then we changed the strategy, we trained discriminator once and followed with 10 times generator training. The loss went well at the beginning, but unstable after few minutes. We noticed that this architecture may not compatible with momentum optimizer, so we changed optimizer to Rmsprop finally and it went well. However, the training process was quite time consuming and it cost days but achieved few results. The original paper mentioned that they use 8 GPUs and trained 2 months to get a roughly satisfied result.

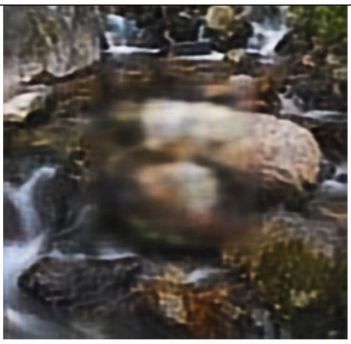
Due to the limited project time and resource we can use. We then changed a little about the training process, we firstly use MSE loss to train the generator trying to make it be able to generate some roughly pieces of contents, and then use WGAN-GP to enhance the results.

c. Modification of convolution padding
In out implemented code, we did not use the zeros padding method to pad every convolutional layer. We used 'REFLECT' padding to replace the default padding strategy. We think reflect padding method can keep information as much as possible for dilate convolutional operation.

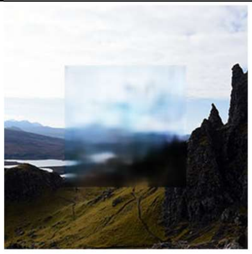
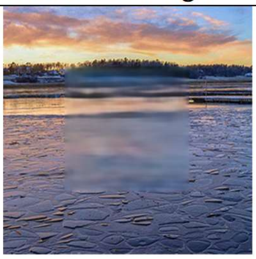
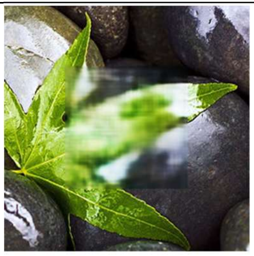
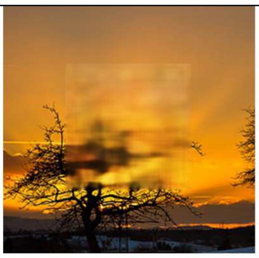
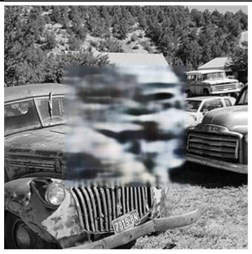
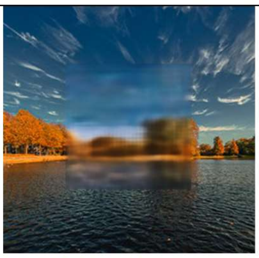
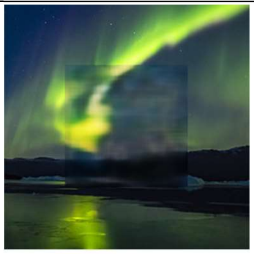
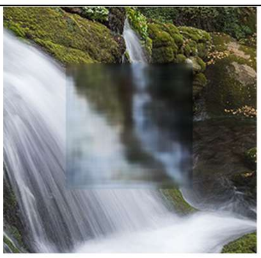
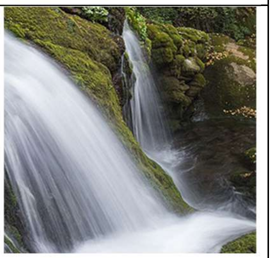
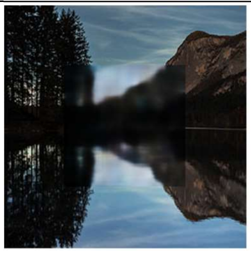
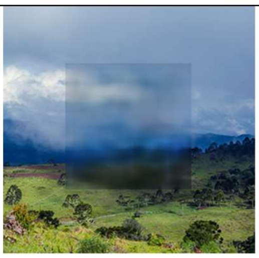
5. Accomplishments

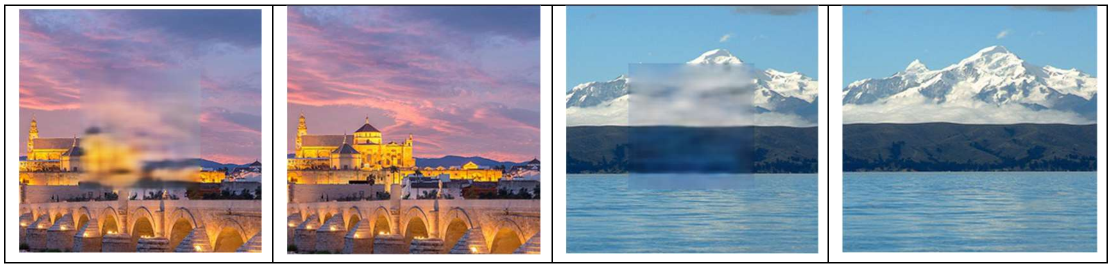
a. MSE pre-training result
We use MSE loss to pretrain the generator for about one day and get result like below:



b. WGAN-GP training result:

Generated images	Original images	Generated images	Original images
			
			
			
			
			



Although MSE pretrained generator only captured roughly contents in original pictures, but it makes generator training much faster in WGAN training process than before. However, the training process based on WGAN is also becoming much slower after around 50 epochs.

6. Future works

a. Due to that we want to simplify the training process and shorten our training time to get some results before project deadline, we fixed the missing mask position in the final modified method. But the ideal situation is that the mask can be anywhere in the input images. Therefore, the first future works should be that randomly choose mask position and mask size to train the network.

b. Due to the input of the generator is an image with blank area in mask region and the generator improves much slower in WGAN-loss optimizer, it is quite hard to generate the best result directly. An alterative way is that split generator into two parts, one is what we have done here to generate a result which is roughly fine but lacks content in detail. The other part is a refine part, use the first stage result to get a better one. The stage 1 result is different from the very beginning's input, it does have content information about mask area, this information is important to guide the stage 2 process to refine the missing parts of image.

7. Reference

[1] Iizuka, S., Simo-Serra, E. and Ishikawa, H., 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, *36*(4), p.107.

[2] Wang, C., Xu, C., Yao, X. and Tao, D., 2019. Evolutionary generative adversarial networks. *IEEE Transactions on Evolutionary Computation*.

[3] Bau, D., Zhu, J.Y., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T. and Torralba, A., 2018. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *arXiv preprint arXiv:1811.10597*.

[4] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. and Huang, T.S., 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5505-5514).

[5] Demir, U. and Unal, G., 2018. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*.

[6] Arjovsky, M., Chintala, S. and Bottou, L., 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

[7] Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.